8. Coding for Error Correction: the Shannon Bound

1. Introduction and Statement of the Theorem

We now consider the problem of transmitting or storing information accurately, when we have noise in our transmission device or dead spots in our storage medium, which destroy some of the information.

Suppose we want to send a message which requires M bits. We mean by this that the actual message will be one among 2^M possible messages that we might send.

We suppose further, that each bit will be in error a certain proportion p of the time. You might imagine p to be 10^{-4} or something like that.

We assume that errors appear independently in our bits with probability p. We have no information, however, as to which bits will be in error when we send our message.

In practice, errors often are not independent, but rather have a tendency to occur in bursts. This is useful information for us, because we can structure our procedures for handling errors so that they are less vulnerable to several errors in a small segment of the message than they are in general. For the moment we will ignore this possibility and assume independence of our errors, in the sense of probability.

The assumptions we have made so far allow us to use both the (weak) Law of Large Numbers derived in the last chapter, and also the binomial distribution of errors that was also discussed.

In this context our plan is to make our message longer than M bits, so that we have some redundancy in it. We want to use that redundancy to recognize the errors in our message, after it is sent and received, or after it is retrieved from storage, and recover our original message.

We will address two questions:

How much redundancy need we add here so that we can recover the original message? In other words, how long must our message actually be?

How can we actually construct useful codes for error correction where the encoding and deocding operations are reasonably efficient?

The answer to the first question is given by **Shannon's Second Theorem**, which is as follows.

Theorem 8.1: To transmit M message bits given a probability p of each bit being in error, for large M requires a proportion H(p) (which is $p\log_2(1/p) + (1-p)\log_2(1/(1-p))$ of additional redundant bits, (called check bits). Thus, if we let the total number of bits needed be N, we have

$$N = M + NH(p).$$

Not only must the code be at least this long, but most (almost all) codes with bit length cM longer than this, that is for any c> 1/(1-H(p)), allow us to recover the message under the given assumptions, with probability approaching 1 as the length of the code goes to ∞. The remainder

of this chapter will consist of proof of this theorem. We will then turn to the second question.

The rest of this section consists of philosophical remarks.

We will find that though Shannon proved his bound in 1948 with a relatively easy argument, for fifty years after this, nobody was been able to discover classes of useful codes for large M and large p ranges, which came come close to this bound. The theorem said, though, that most definable codes with the right length for code words actually work.

The theorem implies that a code in which the code words were chosen at random for each potential message, from among code words of the length noted above, would very probably allow us to recover our message most of the time.

However, finding such a code that could be used in practice was much harder than proving that they existed. It was only in the last decade that codes coming fairly close (*turbo codes* and *low density parity check codes*) were constructed. Even now, nobody can prove that these codes can be made to reach Shannon's bound in the asymptotic limit, although it has been proved that if you tweak the parameters in low density parity check codes in the right way, you can come very close. And for these for two classes of codes, randomization is still used in their constructions, although unlike completely random codes they have efficient algorithms for encoding and decoding them. Nobody knows how to construct codes coming close to Shannon's bound without using randomness.

You might ask, why not generate a code word for each potential message randomly and use that code when you need one?

There are three objections to such a procedure:

First, if M is large, the number of potential messages, 2^M, is far too vast to make an independent choice of code word for each potential message at all possible. OK, we can break our message up into blocks of length k and define a code word for each potential block. This becomes hopeless when the block size gets near 50.

Second, encoding and decoding in such a code requires table look up, which requires keeping the code words for every potential message available. This would require immense storage space.

Finally, decoding when there are errors requires additional work, because the received message with its errors will not be in the table of code words. By the law of large numbers, is will differ from the sent code word by roughly pN
bits. There are techniques for locating the message, (these are extensively used (for example in biology to find genes among sequences of DNA) but they are cumbersome.

Thus, we will actually seek codes which have lots of structure and that make decoding easy even in the presence of errors. For large k and given p it becomes difficult to find codes that both make this possible and meet the Shannon bound.

Finding efficient error correcting codes is a problem that has, then, the remarkable property that human beings have difficulty constructing codes of a given length that are error correcting with proportion of check bits H(p) +c for small c, while almost all randomly chosen codes of this length have the error correcting property.

In other words, for this problem (and quite a few others in computer science) random

constructions are better than anything we know how to devise ourselves.

This tells us a lesson that perhaps explains a bit of history. Throughout the Nineteenth and much of Twentieth Century it was an article of faith among many intellectuals that a planned economy could and should be far more efficient than the random and chaotic economy that is built up from the bottom by the independent action of many individuals.

However, when put into practice in the Twentieth Century, planned economies turned out to be not only quite irksome and occasionally deadly to the intended beneficiaries of the planning, but economic disasters as well.

Perhaps then the construction of an economy is a problem like those of computer science such as finding error correcting codes, where random construction developed locally by a host of individual decisions is almost always better than anything systematic we can come up with imposed from above..

8.2 Proof of the Shannon Bound

We want to prove that we need a proportion of redundant bits beyond our original M bits of at least something like H(p) in order to be able to correctly decode our message.

When we receive a message we will have no idea which bits have been garbled. We will assume, however, that no bits are dropped or added; the only possible mistakes are to switch 0 and 1 here and there throughout the message.

We seek the optimal decoding;
under these circumstances the best candidate to be the true sent message is the message whose code word differs from the received word in the fewest places.

The number of places in which two words of the same length differ is called the **Hamming distance** between them.

The Hamming distance between the true code word and the received word is the number of errors our message was afflicted with. Now we know from the Law of Large Numbers of Chapter 7 that this will be on the order of Np.

If our message is long enough, this number will be quite large, Suppose then that Np + q errors are made, for |q| much smaller than Np.

We now ask, how many different ways are there to make Np+q errors, in a message of length N?

The answer is the binomial coefficient, C(N, Np+q). We have seen in the last chapter that we can write this binomial coefficient as:

$$C(N,Np+q) = 2^{((N*H(p+(q/N)))\ (1+o(1)))}$$

We have 2^M possible sent messages, each of which can accumulate errors in $2^{((N*H(p+(q/N)))\ (1+o(1)))}$ different ways, all of which are equally likely. Suppose we had fewer than $2^{M\ +\ ((N*H(p+(q/N)))\ (1+o(1)))}$
possible received words. Then the pigeonhole principle tells us that some received words must arise

from more than one message. And if there are substantially fewer than this number of received words, then our discussion of probability tells us that on average, each received word will arise from several messages.

By our pigeonhole principle, we need N to be on the order of $\log_2$ of this number of possible received words in order to distinguish among these. For us to have a reasonable chance of decoding then, N must be at least on the order of M+N*H(p).

Which is what we set out to prove.

To summarize the argument, the errors that we know will happen will spread each of our 2^M messages to on the order of $2^{NH(p)}$ possible erroneous messages, and this implies our claim.

8.3 Proof that Random Codes come close to the Shannon Bound.

To prove this, we look at the sample space consisting of all possible codes, (assignments of code words to potential messages) in which all code words have length N, with all possible words having the same probability of being a code word, and code words for different messages chosen independently.

We assume further that each bit of the received word acquires an error with probability p, and that errors on different bits are independent.

Suppose we receive a word R. We know from our weak law of large numbers, that the true decoding of this, the message Z, has a code word C(Z) which has Hamming distance on the order of Np from R.

Our plan will be to decode R to the nearest code word D to it that we find.

If that word is C(Z) we will be right, and otherwise we will be wrong.

Then when will we be wrong?

We will be wrong if there is any other code word D that is closer to R than D is. So we will be right if no other code word has Hamming distance on the order of N(p+ ε) or less from R.

Let the number of words of length N having Hamming distance of at most N(p+ ε) from R be V(N(p+ε))

Since each code word for message words other than R is chosen at random among the 2^N possible code words, it has probability 2^{-N} of being any single word, and probability $V(N(p+\varepsilon))/2^N$ of being within Hamming distance at most N(p+ε) from R.

Thus the probability of that code word being too far from R to be confused with the actual sent code word is at least

$$\mathbf{(1 - V(N(p+ \varepsilon))/2^N).}$$

The probability that this happens independently for each of the $2^M - 1$ other code words is

the product of the probabilities of each of these events happening, or

$$\mathbf{(1 - V(N(p+\varepsilon))/2^N)\ raised\ to\ the\ power\ 2^{M-1}.}$$

And if this happens, we will decode successfully.

Now $V(N(p+\varepsilon))/2^N$
is quite small. One minus it is therefore very close to e raised to the power of $-V(N(p+\varepsilon))/2^N$.

We deduce then, that we will decode successfully with probability at least $\exp(-V(N(p+\varepsilon))/2^N)$ raised to the power 2^{M-1} which is

$$\exp(-V(N(p+\varepsilon))/2^{N-M+1}).$$

Exercise: Prove $\log_2(V(N(p+\varepsilon)))$ is of the order of $NH(p+\varepsilon)$.

By the result you will have proven by performing this exercise, your chance of success in decoding is at least

$$\mathrm{Exp}(-2^{-N+M+NH(p+\varepsilon)}).$$

When N>>M+NH(p+ ε) holds, which means that the number of redundant bits is more than $NH(p+\varepsilon)$, **the exponent here is close to 0 and the probability of successful decoding is close to 1.**

This is what we set out to prove.

If no other code word is in the Hamming sphere of radius $(p+\varepsilon)N$ of R, we will decode our message correctly. That is the long and the short of the argument, which is fleshed out by using probability theory to show that the probability of this approaches one if we have enough check bits.